1.4 Create and Work with Vectors and Matrices in R

```
x <- 11
x
[1] 11

# create vector
> x1 <- c(1,3,5,7,9)
> x1
[1]1 3 5 7 9

# vector of character
gender <- c("male", "female")

> 2:7
[1] 2 3 4 5 6 7

seq(from=1, to=7, by=1)
[1] 1 2 3 4 5 6 7
seq(from=1, to=7, by=1/3)
[1] 1.000000 1.33333 1.666667 2.00000 ...
[9] 3.666667 4.00000 ....
[17] 6.33333 6.666667 ...
seq(from=1, to=7, by=0.25)
[1] 1.00 1.25 1.50 ....

# repeat
rep(1, times=10)
[1] 1 1 1 1 1 1 1 1 1 1
rep("yes", times=5)
[1] "yes" "yes" "yes" "yes" "yes"
rep(1:3, times=5)
[1] 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3
rep(seq(from=2, to=5, by=0.25), times=5)
rep(c("m", "f"), times=3)
[1] "m" "f" "m" "f" "m" "f"

# set x, y vectors
x <- 1:5
[1] 1 2 3 4 5
y <- c(1,3,5,7,9)
[1] 1 3 5 7 9
```

```
x + 10
x - 10
x*10
x/2

# if two vectors of the same length, we may add/subtract/mult/div
# corresponding elements
x + y

# extract specific elements 選取特定
y[3]
[1] 5
y[-3]
[1] 1 3 7 9
y[1:3]
[1] 1 3 5
y[c(1, 5)]
[1] 1 9
y[-c(1,5)]
[1] 3 5 7
y[y<6]
[1] 1 3 5

# matrix 矩陣
mat <- matrix(c(1,2,3,4,5,6,7,8,9), nrow=3, byrow=TRUE)
    [,1] [,2] [,3]
[1,]  1    2    3
[2,]  4    5    6
[3,]  7    8    9
matrix(c(1,2,3,4,5,6,7,8,9), nrow=3, byrow=FALSE)
    [,1] [,2] [,3]
[1,]  1    4    7
[2,]  2    5    8
[3,]  3    6    9

# [列, 欄]
mat[1, 2]
[1] 2
mat[c(1, 3), 2]
[1] 2 8
mat[2,]
[1] 4 5 6
mat*10

# nrow橫列, ncol直行, FALSE照行來排
matrixOne <- matrix(1:100, nrow=10, ncol=10, byrow=FALSE)
       [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
 [1,]    1   11   21   31   41   51   61   71   81    91
 [2,]    2   12   22   32   42   52   62   72   82    92
 [3,]    3   13   23   33   43   53   63   73   83    93
 [4,]    4   14   24   34   44   54   64   74   84    94
 [5,]    5   15   25   35   45   55   65   75   85    95
 [6,]    6   16   26   36   46   56   66   76   86    96
 [7,]    7   17   27   37   47   57   67   77   87    97
 [8,]    8   18   28   38   48   58   68   78   88    98
 [9,]    9   19   29   39   49   59   69   79   89    99
[10,]   10   20   30   40   50   60   70   80   90   100
# Submatrix 子矩陣
matrixSub <- matrixOne[7:8,3:5]
      [,1] [,2] [,3]
[1,]   27   37   47
[2,]   28   38   48

matrixMinus[2,3] <- "try" # 字串取代數字
# 整個matrix變為字串
```

```
> aa <- data.frame(nickname=c("John","Mary","Leo"), weight=60:62, Height=c(160,170,180))
> aa
  nickname weight Height
1     John     60    160
2     Mary     61    170
3      Leo     62    180

# 改變資料型態：改成character
aa[,1] <- as.character(aa[,1])

> View(aa) # 跳出aa data
```

seq序列

```
# seq序列
> ff <- seq(3,9,2)  # 2是間隔
> ff
[1] 3 5 7 9
> ff <- seq(-3,-9,-2)
> ff
[1] -3 -5 -7 -9
```

💡 題目：2 4 6 8 重複3次

```
qq <- rep(seq(2,8,2), 3)
```

💡 放進2列6行的矩陣

```
> qq <- rep(seq(2,8,2), 3)
> qq
 [1] 2 4 6 8 2 4 6 8 2 4 6 8
> mat <- matrix(qq, nrow=2, ncol=6, byrow=TRUE)
> mat
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    2    4    6    8    2    4
[2,]    6    8    2    4    6    8
```

```
> bb <- matrix(1:12, nrow=6)
> bb
     [,1] [,2]
[1,]    1    7
[2,]    2    8
[3,]    3    9
[4,]    4   10
[5,]    5   11
[6,]    6   12
> class(bb)
[1] "matrix" "array"

# 資料型態
> bb <- as.data.frame(bb)
> bb
  V1 V2
1  1  7
2  2  8
3  3  9
4  4 10
5  5 11
6  6 12
> class(bb)
[1] "data.frame"

# colnames 欄位名稱
> colnames(bb) <- c("col1","col2")
> bb
  col1 col2
1    1    7
2    2    8
3    3    9
4    4   10
```

```
5    5   11
6    6   12

# 變回matrix
> bb <- as.matrix(bb)
> bb
     col1 col2
[1,]    1    7
[2,]    2    8
[3,]    3    9
[4,]    4   10
[5,]    5   11
[6,]    6   12
```

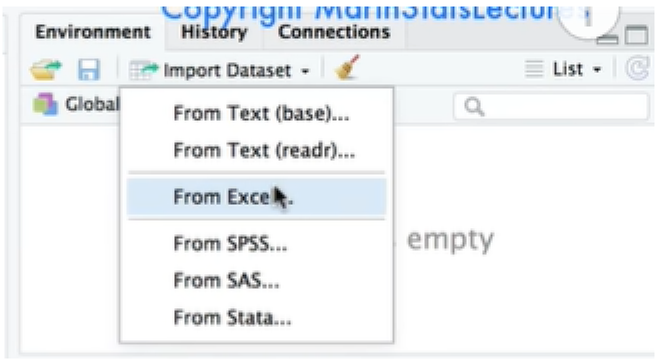
1.5 Import Data, Copy Data from Excel to R CSV & TXT Files

```
# save the file as .csv/.txt
# import file
data1 <- read.csv(file.choose(), header=TRUE)
data2 <- read.table(file.choose(), header=T, sep=",")

data3 <- read.delim(file.choose(), header=T)
data4 <- read.table(file.choose(), header=T, sep="\t")
```

Importing/Reading Excel data into R using RStudio (readxl)

💡 **readxl** package can import both .xlsx & .xls files

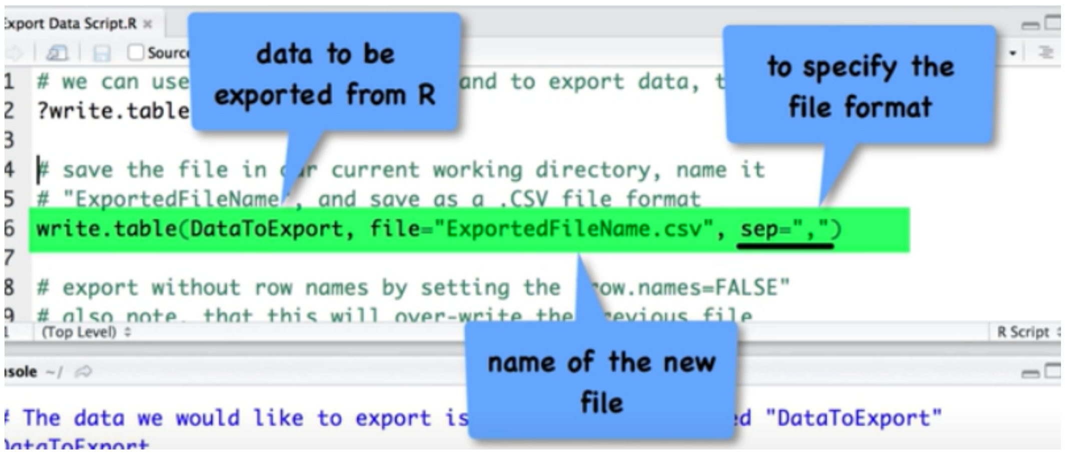
> File - Import Dataset - From Excel



1.6 Export Data from R (csv , txt and other formats)

💡 The most flexible command for exporting data from R is **write.table**

```
?write.table
# save the file in our current working directory, name it
# "ExportedFileName", and save as .CSV file format
write.table(DataToExport, file="ExportedFileName.csv", sep=",")
```



```
# we can use                 and to export data, t
?write.table

# save the file in o r current working directory, name it
# "ExportedFileName", and save as a .CSV file format
write.table(DataToExport, file="ExportedFileName.csv", sep=",")

# export without row names by setting the  ow.names=FALSE"
# also note, that this will over-write the  evious file
```

```
# The data we would like to export is                d "DataToExport"
DataToExport
```

💡 To get rid of the row names while exporting data from R to Excel: **row.names=FALSE**

```
write.table(DataToExport, file="ExportedFileName.csv", row.names=F, sep=",")

# export into a different working directory
# specify the path for where to save the file instead 指定特定路徑
```

```
write.table(DataToExport, file="/Users/..../ExportedFileName.csv", row.names=F, sep=",")
```

💡 To export data from R into a comma separated value file (.csv) use: **write.csv**

```
# don't need sep=","
write.csv(DataToExport, file="/Users/..../ExportedFileName.csv", row.names=F)
```

💡 **write.table** command allows us to save the file in other formats (e.g. tab-delimited text file)

```
# sep="\t"
write.table(DataToExport, file="/Users/..../ExportedFileName.txt", row.names=F, sep="\t")
#  sep=","
write.table(DataToExport, file="/Users/..../ExportedFileName.txt", row.names=F, sep=",")
```

1.7 Importing , Checking and Working with Data in R

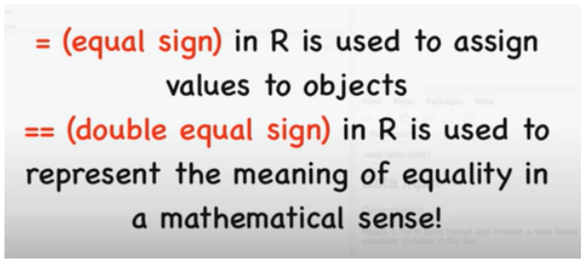
```
# how to input data
> help(read.table)
> ?read.table

> Data1 <- read.table(file="檔案路徑", header=TRUE, sep="\t")
> Data2 <- read.table(file.choose(), header=TRUE, sep="\t")

# remove data
> rm(Data1)
> rm(Data2)

# know the dimensions of the data
> dim(Data1)
[1] 725 6

# first 6 rows
> head(Data1)
# last 6 rows
> tail(Data1)

# 顯示特定幾列
> Data1[c(5,6,7,8,9), ]
> Data1[5:9, ]

# 顯示特定幾列除外
> Data1[-(4:722), ]
# 顯示1, 2, 3, 723, 724, 725

# show the names
> names(Data1)
[1] "LungCap" "Age" "Height" ...
```

1.8 Working with Variables and Data in R

```
# variable names
names(LungCapData) #LungCapData：檔案名
[1] "LungCap" "Age" "Height"

# $ extract variables
1) mean(LungCapData$Age)
2) attach(LungCapData)
   mean(Age)
   detach(LungCapData) # unattach

# class -> factor --> levels
levels(Gender)
[1] "female" "male"

summary(LungCapData)

# convert x(numeric) to a categorical variable/factor
x <- c(0,1,1,1,0,0,0,0,0)
x <- as.factor(x)
```

1.9 Subsetting (Sort/Select) Data in R with Square Brackets

```
dim(LungCapData)
[1] 725 6 # rows, columns

# ==
mean(Age[Gender=="female"])
mean(Age[Gender=="male"])

FemData <- LungCapData[Gender=="female", ]
MaleData <- LungCapData[Gender=="male", ]

MaleOver15 <- LungCapData[Gender=="male" & Age>15, ]
```



= (equal sign) in R is used to assign values to objects
== (double equal sign) in R is used to represent the meaning of equality in a mathematical sense!

1.10 Logic Statements (TRUE/FALSE), cbind and rbind Functions in R

```
Age[1:5]
[1] 6 18 16 14 5
temp <- Age>15
temp[1:5]
[1] FALSE TRUE TRUE FALSE FALSE

temp2 <- as.numeric(Age>15)
temp2[1:5]
[1] 0 1 1 0 0

FemSmoke <- Gender=="female" & Smoke"yes"
FemSmoke[1:5, ]
[1] FALSE TRUE FALSE FALSE FALSE
MoreDta <- cbind(LungCapData, FemSmoke) # add one column
```

1.11 Setting Up Working Directory in R

```
getwd() # 看目前預設位置
[1] "/Users/01dMarin"
setwd("/Users/01dMarin/Desktop/Project 1")

# workspace image file
save.image("xxxxxx.Rdata") # 被存在current working directory

rm(list=ls())
q() # quit

load("xxxxx.Rdata")
load(file.choose())
```

💡 session —> set working directory —> choose directory

1.12 Writing Scripts in R

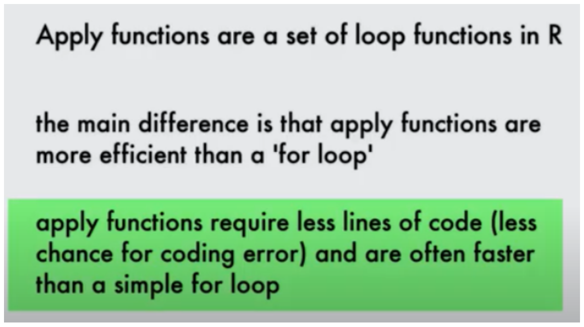
> file —> New —> R script

💡 按tab：快速提示

1.13 How to Install Packages in R

```
help(install.packages)
install.packages("epiR")
install.packages() # list可選

library()
```

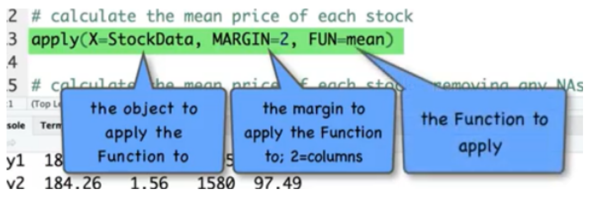
1.15 Apply Function in R

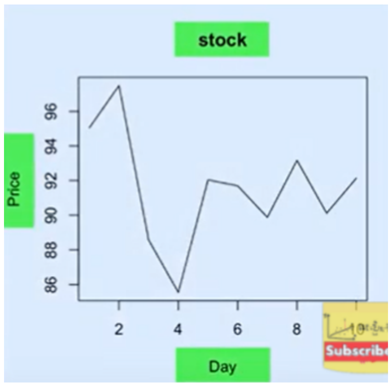

Apply functions are a set of loop functions in R

the main difference is that apply functions are more efficient than a 'for loop'

apply functions require less lines of code (less chance for coding error) and are often faster than a simple for loop

```
?apply
apply(X, MARGIN, FUN,...)
# x-the object we would like to apply some function to
# MARGIN-specifies if fiction applied to rows or columns: 1=row; 2=columns

AVG <- apply(StockData, Margin=2, FUN=mean, na.rm=TRUE)
# 刪除NA

# 欄的mean
colMeans(StockData, na.rm=TRUE)
# find manxnimum
apply(X=StockData, MARGIN=2, FUN=max, na.rm=TRUE)
apply(X=StockData, MARGIN=2, FUN=quantile, probs=c(0.2, 0.8), na.rm=TRUE)

apply(X=StockData, Margin=2, FUN=plot, type="l")
apply(StockData, Margin=2, FUN=plot, type="l", main="stock", ylab="Price",xlab="Day")")
# sum
apply(X=StockData, MARGIN=1, FUN=sum, na.rm=TRUE)
rowSunms(StockData, na.rm=TRUE)
# POINTS
points(apply(X=StockData, MARGIN=1, FUN=sum, na.rm=TRUE),pch=16,col="blue")
```


```
# calculate the mean price of each stock
apply(X=StockData, MARGIN=2, FUN=mean)
```
the object to apply the Function to
the margin to apply the Function to; 2=columns
the Function to apply


stock
Price
Day
96
94
92
90
88
86
2
4
6
8
Subscribe

1.16 tApply Function in R

💡 t-apply can be used to apply a function to subsets of a variable or vector

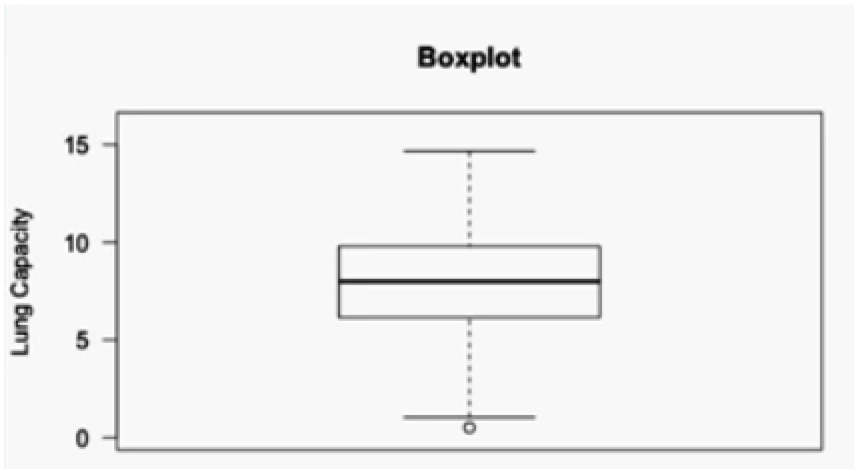
```
tapply(X, INDEX, FUN=NULL,..., simplify=TRUE)

# calculate the mean Age for Smoker/NonSmoker
tapply(X=Age, INDEX=Smoke, FUN=mean, na.rm=T)
m <- tapply(Age, Smoke, mean) # simple vector
      no          yes
12.03549     14.77922
tapply(Age, Smoke, mean, simplify=FALSE) # a list format
$no
[1] 12.03549
$yes
[1] 14.77922

mean(Age[Smoke=="no"])
mean(Age[Smoke=="yes"])
tapply(Age, Smoke, summary)
tapply(Age, Smoke, quantile, probs=c(0.2,0.8))

# subsets based on multiple variables/vectors
tapply(X=Age, INDEX=list(Smoke, Gender), FUN=mean, na.rm=T)
```

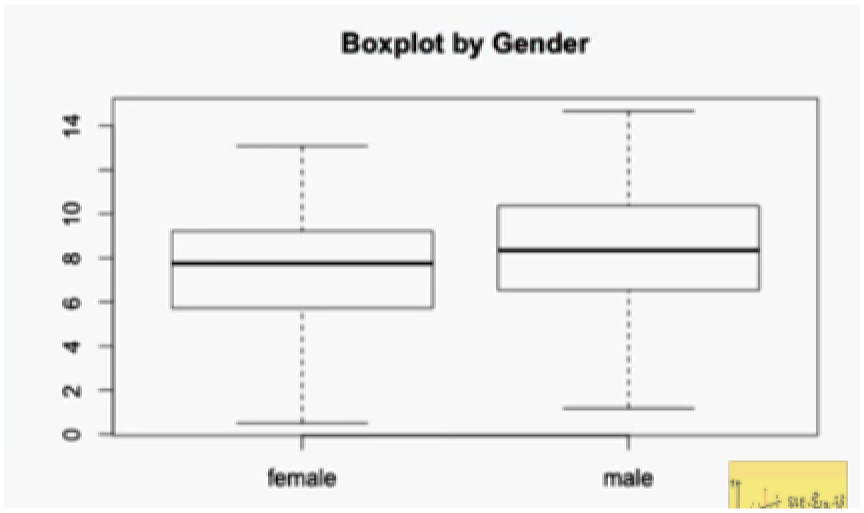
2.1 Bar Charts and Pie Charts in R

```
> LungCapData <- read.table(file.choose(), header=T, sep="\t")
> attach(LungCapData)
> dim(LungCapData)
[1] 725   6
> names(LungCapData)
[1] "LungCap" "Age" "Height" "Smoke" "Gender" "Caesarean"

?barplot
count <- table(Gender)
female    male
   358     367
percent <- table(Gender)/725 # percentage
female       male
  0.4937   0/5062

barplot(count)
barplot(percent, main="TITLE", xlab="Gender", ylab="%", las=1)
barplot(percent, main="TITLE", xlab="Gender", ylab="%", las=1, names.arg=c("Female", "Male"), horiz=TRUE)

pie(count, main="TITLE HERE")

box()
```

2.2 Boxplots and Grouped Boxplots in R

```
> LungCapData <- read.table(file.choose(), header=T, sep="\t")
> attach(LungCapData)

# 盒鬚圖
boxplot(LungCap)

quantile(LungCap, probs=c(0, 0.25, 0.5, 0.75, 1))
boxplot(LungCap,main=boxplot, ylab="Lung Capcity", ylim=c(0, 16), las=1)
boxplot(LungCap ~ Gender, main="Boxplot br Gender")
```




Boxplot by Gender
0
2
4
6
8
10
14
female
male